Analysis Chloe Josien

1. Does the straight-line distance (the absolute distance, ignoring any walls) from the start point to the goal point affect the running time of your algorithm? In other words, does how close the Start and Goal are affect how fast your algorithm finds a shortest "Manhattan" distance?

Yes, because the closer the start and goal are to each other the less time it will take for the algorithm to go through all the possibilities while if they are farther apart it will take more time to go through the algorithm.

2. Explain the difference between the straight-line distance and the actual solution path length. Give an example of a situation in which they differ greatly. How do each of them affect the running time of your algorithm? Which one is a more accurate indicator of run-time?

The straight line distance from the start to the goal ignoring any walls while the actual solution has to go around all the walls using legal steps. One example is pictured below

XXXXXXXXXXXXXX

X X

X X X

X X X

X X X

X SXG X

XXXXXXXXXXXXXX

They differ from each other by the straight line being able to go faster since it can ignore walls but the actual solution has to go around and take more time. The actual solution because it has to solve the maze instead of ignoring the rules.

3. Assuming that the input maze is square (height and width are the same), consider the problem size, N to be the length of one side of the maze. What is the worst-case performance of your algorithm in Big-Oh notation? Your analysis should take into account the density of the maze (how many wall segments there are in the field).

Our algorithm has a big-oh complexity O(N) N is the number of empty spaces in the maze. The worst case for our algorithm is if it was given an empty maze because it would have to search through each and every node for the shortest path.

4. Create a test example to verify your thinking. If possible, augment your code to keep a counter of how often a node is looked at (i.e., every time your code checks to see if a node has already been visited, it is being "looked at").

XXXXXXXXXXXXXX

X S X

X X

X X

X X

X GX

XXXXXXXXXXXXXX

When we counted how many nodes it looked at to solve it counted 260 times for 7 by 16 maze while for a simple maze of the same size it took 164 looks. This shows that our thinking is probable in that an empty maze takes longer to solve than a maze that has more walls.

5. Hypothesize what effect using depth first search would have on the path planning? Provide an example maze for the purpose of this discussion. Would depth first solve the above example in a faster or slower manner?

Depth first search would take longer because depth first search has to go down all the way before backtracking while breadth first search checks on each level each time.

XXXXXXXXXXXXXX

XS G X

X X X

X X X

X X X

X X X

XXXXXXXXXXXXXX

In this example depth first would go through every column taking more time than the breadth first search because the goal is one the same row as the start.

6. One more thought problem (don't spend more that 15 minutes on this). Say you created an entire matrix of nodes representing the maze and did not store any edges. To know if an edge exists for any node at (R,C) you would have to check (R-1,C), (R+1,C), (R,C+1), and (R,C-1). Can you think of another algorithm to find the shortest path from a given start (R1,C1) to a goal (R2,C2).

One algorithm I can think of is a brute force algorithm. Similar to the sudoku solver the algorithm would guess a direction until it either finds the goal or hits a dead end and has to backtrack to find a previous guess and try a different path.

7. How many hours did you spend on this assignment?

I spent about 4 hours.